4A

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: 0 100 821 A2

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 83104483.9

(22) Date of filing: 06.05.83

(51) Int. Cl.³: G 06 F 15/40

(30) Priority: 30.06.82 US 393902
21.06.82 US 390163
21.06.82 US 390454
21.06.82 US 390485
24.06.82 US 391629
29.06.82 US 393244

(43) Date of publication of application: 22.02.84 Bulletin 84/8

(84) Designated Contracting States: DE FR GB IT

(71) Applicant: International Business Machines Corporation
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: Haderle, Donald James
812 Lilac Way
Los Gatos California 95030(US)

(72) Inventor: Shan, Ming-Chien
6337 Slida Dr.
San Jose California 95129(US)

(72) Inventor: Putzolu, Gianfranco Raimondo
17550 Nillview Ave.
Morgan Hill, Ca 95037(US)

(72) Inventor: Traiger, Irving Lawrence
1031 Mt. Darwin Dr.
San Jose CA 95120(US)

(74) Representative: Ahlman, Bertel
IBM Svenska AB Box 962
S-18 109 Lidingö(SE)

(54) Method and apparatus for managing a database.

(57) A computing system is characterized by a database management system which provides method and apparatus for online definition of data descriptors (77) while maintaining the availability of data (76) and data descriptors not impacted by a concurrent definitional process. Under the control of the database management system (70) of the invention, the domain effected by the definitional process is determined, transactions which reference descriptors within the domain effected by the definitional process are marked for rebinding, prior-to-commit access by transactions which have not been prebound to objects within the domain effected by the definitional process is inhibited, and thereafter all accesses to objects in the domain are allowed.

EP 0 100 821 A2

./...

Croydon Printing Company Ltd.

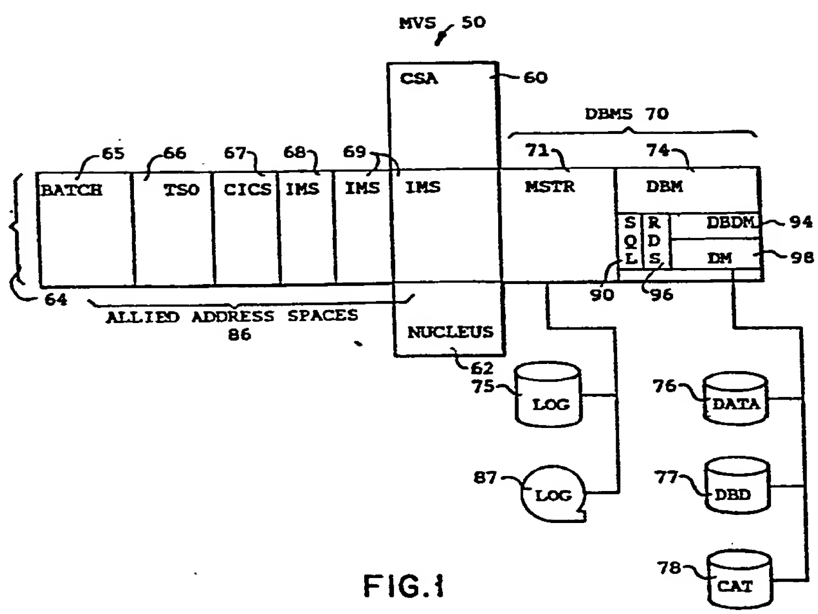

MVS 50
CSA
60
DBMS 70
65
66
67
68
69
71
74
BATCH
TSO
CICS
IMS
IMS
IMS
MSTR
DBM
S Q L
R D S
DBDM
94
DM
98
64
90
96
ALLIED ADDRESS SPACES
86
NUCLEUS
62
75
LOG
76
DATA
87
LOG
77
DBD
78
CAT

FIG.1

0100821

Method and apparatus for managing a database.

This invention relates to a new computing apparatus and method for managing a database. More specifically, it relates to an apparatus and method for executing user transactions in a data base computing system.

Computing systems typically include a collection of data which is referenced by application programs. A database management system is provided for managing the creation, modification, and use of such data collections and their descriptors.

In one prior art database management system, the IBM IMS/VS Version 1, in order to make changes in data descriptors, a database administrator (DBA) first generates a new set of descriptors, such as physical/logical database descriptors (DBDs) offline (that is, indepedently of the installed IMS system). The DBA then invokes an IMS utility to unload the data on tapes, replace the descriptors in an IMS DBD library, and then invoke the IMS utility to reload the data. Of course, during the definitional (unloading and reloading) operations, the data is not available for access by application programs or transactions.

In prior art relational database management systems, such as the IBM Sequel Query Language (SQL), the database and its associated descriptors comprise one or more logical spaces called segments. In a relational database management system of this type, one or more transations may be prebound to reference certain descriptors. Any transaction requiring a

descriptor change is required to lock the entire segment in which the target descriptor resides. This approach provides a certain degree of dynamic definitional change at the expense of prohibiting all transactions from referencing any data which is not subject to change but resides in the same segment.

Consequently, there is a need in the art for a database management system having enhanced data definitional capabilities which provide high availability of operational data and low user operational overhead.

The present invention is defined in the attached claims.

The invention provides a computing system characterized by a database management system which provides for online definition of data descriptors while maintaining the availability of data and data descriptors not impacted by a concurrent definitional process. Under the control of the database management system of the invention, the domain effected by the definitional process is determined, transactions which reference descriptors within the domain effected by the definitional process are marked for rebinding, prior-to-commit access by transactions which have not been prebound to objects within the domain effected by the definitional process is inhibited, and thereafter all accesses to objects in the domain are allowed. Under the control of the database management system of the invention, the domain effected by the definitional process is determined, transactions which reference descriptors within the domain effected by the definitional process are marked for rebinding, prior-to-commit access by transactions which have not been prebound to objects within the domain effected by the definitional process is inhibited, and thereafter all accesses to objects in the domain are allowed.

The invention will now be described with reference to the attached drawings.

Figure 1 is a diagrammatic illustration of a computing system including a database management system.

Fig. 2 is a diagrammatic illustration of the processing of a data definition language statement through various components of the database manager of Figure 1.

Figure 3 is a diagrammatic illustration of the processing of a data manipulation language statement through various components of the database manager of Figure 1.

Figure 4 is a diagrammatic illustration of a database descriptor.

Figure 5 is a diagrammatic illustration of a database descriptor section.

Figure 6 is a diagrammatic illustration of the processing of an application program containing Sequel statements into executable text.

Figure 7 is a diagrammatic illustration of a SKCT, or plan, showing its relationship to a database descriptor.

DESCRIPTION OF THE PREFERRED EMBODIMENT

In Table 1 is set forth a listing of the abbreviations used in the following description of a preferred embodiment of the invention.

TABLE 1 ABBREVIATIONS LISTING

CL	Control Language
CT	Cursor Table
CTDB	Cursor Table Database Descriptor Block
CUB	Cursor Block
DBA	Database Administrator
DBD	Database Descriptor
DBTG	Database Task Group
DBDM	Database Descriptor Manager
DBRM	Database Request Module
DDL	Data Definition Language
DM	Data Manipulation Component
DML	Data Manipulation Language
OBD	Object Descriptor
OBID	Object Identifier
PSCB	Page Set Control Block
RDS	Relational Data System
SKCT	Skeleton Cursor Table
SQL	Sequel Query Language

Referring to Figure 1, a storage map is illustrated for a typical IBM MVS system 50, including common system area (CSA) 60, nucleus 62, and a collection of private address spaces 64 comprising batch address space 65, TSO address space 66, CICS address space 67, IMS/VS address spaces 68, 69, and database management system (DBMS) 70 address spaces master (MSTR) 71 and database manager (DBM) 74. DBMS MSTR 71 includes modules providing control, recovery, and logging functions. The control modules interface the application programs in allied address spaces 86 to DBMS 70. The logging modules maintain the system logs on DASD 75 and archive tape 87, and the recovery modules utilize those logs in restoring the system to operation following an interruption. These facilities are more fully described in copending

EP patent applications by D. Gawlick et al for "Method and Apparatus for Logging Journal Data Using a Log Write Ahead Data Set", number ____________, applicants reference number SA982004, by E. Jenner for "Method and Apparatus for Restarting a Computing System", number ____________, applicants reference number SA981047, and by R. Reinsch for "Computing System and Method for Operating the same", number ____________, applicants reference number SA981052. Data base manager 74 includes modules controlling access to data repository 76, utilizing data descriptors 77 and catalog 78. DBM 74 modules include relational data system 96, data manipulation component 98; and database descriptor manager 94. Allied address spaces 86 include the application programs written in a language such as PL/I which, as will be more completely described in connection with Figure 6, contain Sequel Query Language (SQL) statements for defining and manipulating data in data resource 76. Application programs 86 communicate to the DBM 74 via the language SQL which has two component parts: a Data Definition Language (DDL) component and a Data Manipulation Language (DML) component.

Operating system 50 may be an IBM OS/MVS (as shown) system executing on an IBM System/370. The IBM System/370 is described in IBM System/370 Principles of Operation, IBM Publication GA22-7000-6.

Referring to Figure 2, the processing of a data definition language statement is illustrated. DDL statements of interest to the present invention include the following:

CREATE
ALTER
DROP

Each of these statement reference a data 76 object. In Table 2 is set forth the terms used by users of SQL 90 for referring to these objects together with the corresponding internal descriptors created by DBDM 94.

TABLE 2 DATA OBJECTS 76

SEQUEL DDL TERMS	DBDM INTERNAL DESCRIPTORS
DATABASE	DATABASE
TABLE SPACE	FILE FILE PAGE SET
TABLE	RECORD TYPE
INDEX	FAN SET INDEX PAGE SET

VSAM data set(s) are created or associated with file page sets or index page sets, and provide the actual space where data supplied by DML statements will be stored.

During a bind process, a DDL statement 100 is passed from an application 101 address space 86 to RDS 96 which first passes a request to the data manipulation component (DM) 98. DM 98 records in database catalog 78 a description of statement 100 which is in a form accessible for external access. Next, the request is passed by RDS 96 to DBDM 94, the DBM 74 module which

controls the creation, reading, deletion, and alteration of data descriptors. DBDM 94 establishes internal descriptors required to satisfy this statement 100 which it passes to DM 98 for recording on DBD 77, a database containing internal descriptors in a compact form for efficient retrieval. DBD 77 will be more fully described in connection with Figures 4 and 5.

Referring now to Figure 3, a data manipulation language (DML) statement 102 is processed as shown. DML statements of interest to the present invention include the following:

SELECT
INSERT
UPDATE
DELETE

Each of these DML statements 102 reference only rows (or tuples) within a table described by a DDL statement 100. A DML statement 102 from some allied region 86 is passed to RDS 96 which first access DBD 77 through DM 98 to obtain the descriptor for the data object specified in statement 102, and then accesses the data object in data set 76, again using the services of DM 98.

For further information with respect to relational data systems of the type herein described, reference is made to G. Sandberg, "A primer on relational database concepts", IBM System Journal, Vol. 20, No. 1, 1981, pp. 23-41, and to M. W. Blasgen, et al, "System R: An architectural overview", IBM System Journal, Vol 20, No. 1, 1981, pp. 41-62.

Referring now to Figures 4 and 5, a description will be given of DBD 77. Either DBD 77 or an OBD 122-128 can be used as the basic unit in definitional operations. A DBD 77 comprises one or more sections 110-114 which are

chained together, to accommodate changes while keeping pointers valid. Each DBD section 110 includes header 120, object ID (OBID) map 121, contiguous free area 129, and an allocated area including OBDs 122-128 and holes 130. Header 120 contains a pointer to the next DBD section 112 in DBD 77, the offset to contiguous free area 129, the total amount of contiguous free space, and the present number of entries in OBID map 121. OBID map 121 is a list of, say, four byte entries containing offsets from the beginning of the DBD 77 to the OBDs 122-128 in the allocated area, or zeros for available OBIDs 121. The list is addressed using two byte IDs which are the OBIDs of the corresponding OBDs 122-128. OBIDs 121 are reusable after commitment of OBD 122-128 delete operations. Holes 130 are spaces resulting from OBD 122-128 deletions. Each OBD describes a data object 140, 142, 144 on database 76. As shown, OBD 124 describes data object 140. Each OBD 122-128 may be referenced by an SKCT 150 plan, which will be more fully described in connection with Figure 7. As shown in Figure 5, PLAN A 152 references OBD 124, PLAN B 154 references OBD 122, and both PLAN C 156 and PLAN D 158 reference OBD 126.

Referring now to Figure 6, a description will be given of the compilation of a typical application program, such as application program 153 from allied address space 86 which includes, for example, a SQL SELECT statement. Precompiler 160 precompiles the application code 153 into modified application code 162 which replaces the SQL statements in code 153 with calls including appropriate parameter lists to database request module 180 in DBMS 70. Compiler 170 compiles the modified code 162 to produce executable text 174 which is then link edited into an execution load module 177. Precompiler 160 extracts the SQL statements from the application code 153 and places them in a file called a Database Request Module 180. These are bound 184 by a

process to be described hereafter in connection with Figure 7 into a skeleton cursor table (SKCT) 152. SKCT 152 is converted into cursor table (CT) 155 by create thread process 176, as will be more fully described in connection with Figure 7. This process is still further described in Blasgen, et al, supra.

Bind process 184 takes as input a set of database request modules (DBRMs) 180 as specified by a bind processor, saves DBRMs 180 in catalog 78 and names the set. That set, together with other information provided with the DBRMs 180, is referred to as the bind source. Binding also yields the SKCT, or plan, 152 which is the form of the bind source which is internal to DBMS 70. During DDL 100 operations, plans may be deleted, the bind source remains intact, such that it may be rebound when a deleted plan is required.

Referring now to Figure 7, a description will be given of an SKCT, or plan, 152. Plan 152 identifies the set of SQL (DML 102) statements that the corresponding users 150 may execute. Plan 152 includes a header 200 having pointers 202, 204 to sections 210 and 240 of the plan in data manipulation component 98 and relational data system component 96, respectively. DM section 210 includes a plurality of cursor blocks (CUBs) 220 and PSCBs 230. As shown, CUB 220 includes a pointer field 222 identifying the OBD 124 in DBD 77 which describes data object 140. A plurality of such CUBs point to various OBDs describing all of the data objects 76 available to users of this plan 152. RDS section 240 includes a program table 250 providing pointers 254 for each identified bound program 252. As shown, pointer 256 references section lookup table 260 which includes an entry for each section of the bound program. An entry includes the section identifier 262 and a pointer 264 to

the corresponding section. Herein, pointer 266 references section 272, and pointer 268 references section 270. Section 270, by way of illustration, includes the executable code 280 for one DML statement and invocation structures including pointer 274 to the CUB 220 for the corresponding OBD 124.

During an early descriptor bind operation, when this plan 152 is built, RDS 76 obtains from catalog 78 the OBID 276 for the OBD 124, and plugs into CUB field 222 that OBID value. Thereafter, when this plan 152 is referenced on behalf of an authorized user, CUB field 222 is used to search the descriptor OBD 124 for the data object 140 to be referenced and plugs the address of the OBD 124 in CUB field 222.

The existence of a plan 152 requires that the OBDs 124 it references stay unchanged.

During allocation time (create thread 176), skeleton cursor table (SKCT) 152 is copied by DBM 74 and transformed into a cursor table (CT) 155 completing or modifying selected fields. Hereafter, reference to an SKCT may be to the SKCT before allocation or to the CT resulting from allocation.

Locks are used by data manipulation component 98 to serialize the search of OBDs 122-128 to prevent other users from updating them during that search time. So, when updating OBDs 122-128, DBDM 94 obtains an exclusive lock on DBD 77 to prevent searches of the descriptors 122-128 by DM 98. The procedure for changing OBD 126, therefore, is as follows:

1) RDS 96 uses a dependency list created during bind by RDS 96 and stored in catalog 78 to quiesce by way

of a lock on SKCTs 156 and 158 referencing that OBD 126 all early bound users of data structures 142.

2) DBDM 94 locks the DBD 77 containing the OBD 126 exclusive, and makes the change to OBD 126. Alternatively, DBDM 94 could lock at the OBD 126 level.

Even though a lock on DBD 77 is held by DBDM 94, only plans 156 and 158, which reference OBD 126 need be quiesced since only they depend on that structure which is undergoing change.

In the descriptions which follow of the locking protocols, reference will be made to the following locks: (1) a manual lock, which terminates under control of the owner of the resource being accessed; (2) a commit lock, which terminates when the user has reached sync point; and (3) an allocation, create thread, or deallocation lock, which terminates at deallocation time, i.e., when the thread is destroyed. Allocation occurs at create thread 176, when an attempt is made to get the resources needed to execute a plan.

Having given a general description of Figures 1 - 7, the operation of these structures and procedures in carrying out the invention will be explained.

In a relational database 76 of the type managed by the database management system 70 of this preferred embodiment of the invention, data objects 76 with which application program can deal are data bases, tablespaces, tables, and indexes. Associated with each such data object 76 is a database descriptor 77. Each of these data objects 76 can be created, dropped, or altered via appropriate DDL 100 transaction statements .

For each request for a tablespace object 76 creation via statement 100, there are a file descriptor and a file page set descriptor built within database descriptor 77 to represent the tablespace. Similarly, an index descriptor and an index page set descriptor are built in DBD 77 for an index object 76, and a record type descriptor is built to represent a table object 76.

For frequently repeated transactions 103, an application plan (SKCT) 150 is built during a static binding time. The information in plan 150 is used to avoid an expensive binding step at each subsequent execution of transaction statement 102. During the process of static binding, a SQL statement 102 in application program 103 is bound to data object descriptors 76 necessary to satisfy the requirements of the statement. The binding information is recorded in an application plan 152, and the dependency relationships between data objects 76 and plans 152 are stored in catalog 78. Data base management system (DMBS) 70 maintains catalog 78 by inserting new plans 150 when a new relationship is established, and modifying or deleting a plan 150 when the relationship is changed or dropped. This information is also used by DBMS 70 to identify plans 150 that need to be locked to quiesce activity on data objects 76. During allocation, the associated plan 150 is referenced to actually allocate the data objects 76 to the transaction.

For dynamic SQL statements 100, 102 the binding of the actual statement is deferred until execution time, when the transaction dynamically constructs the statement. At this time, a dynamic binding function is invoked to build cursor blocks (CUB) 220 and allocate data objects 76.

Before any change can be made to a data descriptor 122-128, DBMS 70 must assure (1) that no transaction currently running depends upon that descriptor 122-128 (that is, an operation initiated in executing a DDL statement 100 will be denied if there is an outstanding DBMS utility job referencing the target descriptor 77), and (2) that all existing pre-bound plans 150 and internal control blocks 220, 230 affected by the change are invalidated to prevent obsolete information in data objects 76 from being used in the future. In addition, in order to preserve the data integrity of the system, certain DBMS 70 operations will also be denied if they aim at an incomplete or logically deleted data object 76.

The manner in which the various control structures and data areas above described are used to support concurrent, online definitional operations will next be described.

In all cases, the entire DBD 77 is locked in either shared or exclusive mode during DBDM 94 operations, since DBD 77 is the unit of reading and writing descriptors from/to DASD 76. In the case of DBD 77 deletion, DBD 77 is locked in exclusive mode with deallocation duration. This deallocation duration lock is used to prevent other concurrent transactions from seeing the deleted DBD 77. Otherwise, other transactions can locate and access the deleted DBD 77, since DBMS 70 does not erase the virtual memory of a statically bound DBD 77 until deallocation. For other operations, DBD 77 is locked in exclusive mode with commit duration, to insure that backout is possible without impacting operations by other transactions Backout is described in the copending Jenner application previously mentioned. For read operations, DBD 77 is locked in share mode with commit duration, to avoid dirty

reads, i.e. to prevent one transaction from seeing uncommitted DBD 77 changes made by another concurrently running transactions.

It is required that DBD 77 be locked at least in share mode, for at least commit duration, during static and dynamic bind operations, in order to avoid dirty reads of uncommitted descriptor changes.

Since allocation does not acquire locks on DBD 77, synchronization is needed during DDL operations to prohibit pre-bound work units from accessing descriptors 77 while descriptors 77 are changing. In some cases, the definitional subcomponent DBDM 94 of DBMS 70 will lock file page sets 76 to prohibit allocation of data or index page sets whose descriptors 122-128 are undergoing change. There are other cases where page set locking is not sufficient and the related plan 150 must be locked. For example, allocation follows page set OBD 122-128 chains to locate page set descriptors before locking the page sets.

Before describing the detailed locking protocols for all DDL 100 operations, the implication of each general lock category will be described.

A DBD 77 lock in exclusive mode will serialize with binding or other DDL 100 operations within the DBD 77 by other transactions.

A DBD 77 lock in share mode will serialize with DDL 100 operations within the DBD by other transactions.

An SKCT 150 lock in exclusive mode will serialize allocation by other transactions bound to that SKCT.

A lock on a page set 76 will serialize allocation by other transactions to the page set.

A lock on the open function for a page set 76 in exclusive mode will serialize open and close operations on the page set by other transactions.

A DBD 77 is locked before the quiescing of SKCTs 150 in order to prevent new binding to the target object 76. Otherwise, if SKCT 150 quiescing is done first, a new SKCT 150 may be bound to the target object 76 by another transactions in the time interval starting at the completion of locking on existing related SKCTs 150 and ending at the granting of the lock request on the DBD 77. If this occurs, the newly created SKCT 150 would not be quiesced.

Locking of SKCTs 150 must be in exclusive mode for at least commit duration, to prohibit allocation of the given SKCT 150 by other transactions.

In the case of DBD 77 or OBD 122-128 deletion, the lock on the current SKCT 150 must be held until deallocation time. Otherwise, after the next commit of an invoking transaction A, his cursor table (CT) is still active, but has no SKCT 150 to cover it. Therefore, another transaction B may cause definitional changes that impact transaction A, but there is no SKCT 150 to quiesce activity by transaction A.

In general, the locking strategy of the invention favors locking of page sets 76 over the locking of SKCTs 150, since fewer locks are involved. However, where the SKCT 150 must be locked or deleted anyway and where OBD 77 attributes are examined or an OBD 77 chain is followed during allocation before the page sets are locked, the SKCT lock is chosen.

During UNDO processing (as described in the above referenced copending Jenner application) of DDL 100 CREATE and certain ALTER operations, the CTDB 231, PSCB 230, and CUB 220 run time control blocks are marked invalid since the OBD pointers 222 in those control blocks are no longer valid. The invalidation is applied only to those control blocks which are associated with the invoking transaction when the objects referenced by these control blocks have been modified or deleted. Control blocks for other transactions do not need to be invalidated, because the locking protocol insures that only the invoking transaction could still be operating on the referenced objects 76.

Whenever a database 76 is deleted, the corresponding CTDB 231 is marked invalid.

All record 76 CUBs 220 and index 76 CUBs 220 defined on a record type are marked invalid when the record type, file, or database descriptor 77 is deleted, while only index 76 CUBs 220 are marked invalid when an index 76 is deleted. In general, DDL 100 ALTER operations do not affect any CUBs 220, except for the case of addition of new fields or database 76 validation procedure to an existing record type. This is because the record type OBD 122-128 generally has to be moved to tolerate the expansion of the OBD. However, a pointer to the OBD is kept in all related record type and index CUBs 220, so that the CUBs must be marked invalid.

All file page sets 76 PSCBs 230 and all associated index page set 76 PSCBs are marked invalid when a file page set, file, or database 76 is deleted, while only the index page set PSCB 230 is marked invalid when an index page set 76 is deleted.

Having described the various modules and control blocks, the procedures executed to perform on-line definitional change (that is, changes to DBDs 77 and OBDs 122-128) will be explained.

Binding of an Application Plan

In binding an application plan, no DBD 77 lock is needed since a page lock on the catalog table 78 is sufficient for synchronization purposes of a current transaction with other interest-conflict concurrent transactions.

The procedure is:

1) Lock the page in catalog 78 which contains the plan in exclusive mode with commit duration.

2) Lock the target plan 150 in exclusive mode with commit duration.

3) Invoke RDS 96 to process DBRMs 180 , produce SKCT 152 sections 210, 240.

4) Store the target plan 150.

Execution of a Pre-bound SQL Statement

In executing a pre-bound SQL statement 100, 102, an SKCT 150 lock is needed to prevent the plan 150 from being deleted or modified by another transaction and a page set or page 76 lock is needed to synchronize concurrent read/update on the same data items 76 among different transaction.

The procedure is:

1) Lock the associated plan 150 in shared mode with commit duration.

2) Lock the target page set or page 76 in shared or exclusive or update mode with commit duration.

3) Execute the SQL statement 100, 102.

Execution of a Dynamic SQL Statement

In executing a dynamic SQL statement, a DBD 77 lock is needed to prevent the DBD from being deleted or modified by another transaction, and a page set or page lock is needed to synchronize concurrent read or update on the same data items 76 among different transactions.

The procedure is:

1) Lock the target DBD 77 in shared mode with commit duration.

2) Lock the target page set or page 76 in shared or exclusive or update mode with commit duration.

3) Execute the SQL statement 100, 102.

Execution of an Utility Job

In executing a utility job, a DBD 77 lock is needed to allow the utility to update appropriate flags in the DBD, and a page set or page 76 lock is needed to synchronize concurrent read or update on the same data items 76 among different transactions.

The procedure is:

1) Lock the target DBD 77 in exclusive mode with commit duration.

2) Set the "utility-in-process" flag in DBD 77.

3) Lock the target page set or page 76 in shared or exclusive or update mode with commit duration.

4) Do the utility process.

0100821

Execution of a Definitional (DDL) Statement

The remaining procedures set forth the locking protocol for each of the DDL statements 100 of interest to this invention.

Creating a Database Descriptor

In creating a database 76 descriptor 77 a lock on the module which generates database identifiers is used to serialize the allocation of DB identifiers (DBid); however, no DBD 77 lock is needed since the DBD 77 being created has not yet been exposed to another transaction, no SKCT 150 lock is needed since no new SKCT can be bound to the database 76 and SKCTs 150 bound to previous incarnations of the same DBid would have been deleted, and no page set 76 lock is needed since no page set has yet been created in the DBD being created.

The procedure is:

1) Lock the DBid generator in exclusive mode with commit duration.

2) Allocate a DBid to the new DBD 77.

3) Build the new DBD 77.

Deleting a Database Descriptor

In deleting a DBD 77, an SKCT 150 lock is needed to quiesce all concurrent statically bound transactions against any data object in the DBD 77. The locks must be held for at least commit duration, except the lock on SKCT 150 of the current transaction, which must be held for dallocation duration. An SKCT 150 deletion is needed

to stop all subsequent activities against any object in the DBD 77.

The procedure is:

1) Lock the target DBD 77 in exclusive mode with deallocation duration.

2) Lock the plans 150 on all files in DBD 77 in exclusive mode with commit or deallocation duration.

3) Delete the plans 150 on all files in DBD 77.

4) Invalidate the CTDB 231 and all record type or index 76 CUBs 220 and all page set cursor blocks (PSCBs) 230 on any objects in DBD 77.

5) Drop the DBD 77.

Creating a File Descriptor

In creating a file 76 descriptor 77, no SKCT 150 lock is needed since no SKCT can yet be bound to the file, and no page set 76 lock is needed since no page set has yet been created in the file 76.

The procedure is:

1) Lock the target DBD 77 in exclusive mode with commit duration.

2) Build the file 76 OBD 124 in DBD 77.

Deleting a File Descriptor

In deleting a file 76 descriptor 77, an SKCT 150 lock is needed to quiesce all concurrent statically bound activities against any object in file 76. The locks are held for at least commit duration, except the lock on the SKCT 150 of the current transaction, which must be held for deallocation duration. Deletion of an SKCT 150 is needed to stop all subsequent activities against any object in the file.

The procedure is:

1) Lock target DBD 77 in exclusive mode with commit duration.

2) Lock the plans (SKCTs) 156, 158 on the target file 142 in exclusive mode with commit or deallocation duration.

3) Delete the plans 156, 158 on the target file 142.

4) Invalidate all record type or index 76 CUBs 220 and all page set PSCBs 230 on the target file 142.

5) Drop the file 142 OBD 126 from the DBD 77.

<u>Altering a File Descriptor</u>

In altering a file 140 descriptor no lock on SKCT 152 is needed since no prebound plan 152 will be affected by any allowable updates in file 140 OBD 124.

The procedure is:

1) Lock the target DBD 77 in exclusive mode with commit duration.

2) Lock the page set 76 associated with the target file 140 in exclusive mode with commit duration.

3) Alter the target file 140 OBD 124 in DBD 77.

Creating a Record Type Descriptor

In creating a record type 76 descriptor 77, no SKCT 150 lock is needed since no SKCT 150 can yet be bound to it.

The procedure is:

1) Lock the target DBD 77 in exclusive mode with commit duration.

2) Build the record type 76 OBD 128 in DBD 77.

Deleting a Record Type Descriptor

In deleting a record type 142 descriptor 77, an SKCT 156, 158 lock is need to quiesce all concurrent statically bound activities against the record type. The locks are held for at least commit duration, except the lock on the SKCT 150 of the current transaction, which must be held for deallocation duration. An SKCT 156, 158 deletion is needed to stop all subsequent activities against the record type.

The procedure is:

1) Lock the target DBD 77 in exclusive mode with commit duration.

2) Lock the plans 156, 158 on the target record type 142 in exclusive mode with commit or deallocation duration.

3) Delete all plans 156, 158 on the record type.

4) Invalidate the record type 142 and all index 140 CUBs 220 on the target record type 142.

5) Drop the record type 142 OBD 126 from the DBD 77.

Altering a Record Type Descriptor

In altering a record type 142 descriptor 77, an SKCT 150 lock is needed since no lock is issued on a page set 76 for a partitioned file at allocation time. The locks must be held for at least commit duration, except the lock on the SKCT of the current transaction, which must be held for deallocation duration.

The procedure is:

1) Lock the target DBD 77 in exclusive mode with commit duration.

2) Lock the plans 156, 158 on the target record type 142 in exclusive mode with commit or deallocation duration.

3) Invalidate the record type and all index CUBs 220 on the target record type.

4) Alter the target record type OBD 126 in DBD 77.

Creating an Index Descriptor

In creating an index 140 descriptor 77, no SKCT 150 lock is needed since no SKCT could have been bound to the index 140. The page set 142 lock is used to guarantee the

consistency of the index chain maintained in DBD 77, which will be followed by DBMS 70 during SQL 90 updating and index maintenance.

The procedure is:

1) Lock the target DBD 77 in exclusive mode with commit duration.

2) Lock the page set 76 associated with the target file in exclusive mode with commit duration.

3) Build the target index OBD 124 in DBD 77.

Deleting an Index Descriptor

In deleting an index 140 descriptor 77, an SKCT 152 lock is needed to quiesce all concurrent transactions having statically bound CUBs 220 on the index. The locks are held for at least commit duration, except the lock on the SKCT 150 of the current transaction, which must be held for deallocation duration. The SKCT deletion is needed to stop all subsequent explicit use of the index 140. The page set lock is used to guarantee the consistency of the index chain maintained in DBD 77, which will be followed by DBMS 70 during SQL updating and index 140 maintenance.

The procedure is:

1) Lock the target DBD 77 in exclusive mode with commit duration.

2) Lock the plans 152 on the target index in exclusive mode with commit or deallocation duration.

3) Delete all plans 152 on the index 140.

4) Lock the page set 76 associated with the target file in exclusive mode with commit duration.

5) Invalidate all index CUBs 220 on the target index.

6) Drop the index OBD 124 from the DBD 77.

Altering an Index Descriptor

In altering an index 140 descriptor 77 there is no index page set lock provided. The lock on an index page set is achieved by locking the corresponding file page set. No lock on plan 152 is needed since no prebound plan will be affected by any allowable updates in the index 140 OBD 124.

The procedure is:

1) Lock the target DBD 77 in exclusive mode with commit duration.

2) Lock the page set 144 associated with the target file in exclusive mode with commit duration.

3) Alter the target index OBD 124 in DBD 77.

Creating a File Page Set Descriptor

In creating a file page set 140 descriptor, an SKCT 152 lock is used to protect the file page set chain which is followed during DBMS 70 allocation processes. The locks are held for at least commit duration, except the lock on the SKCT of the current transaction, which must be held for deallocation duration.

The procedure is:

1) Lock the target DBD 77 in exclusive mode with commit duration.

2) Lock the plans 150 on the associated file 144 in exclusive mode with commit or deallocation duration.

3) Build the target file page set 140 OBD 124 in the DBD 77.

Deleting a File Page Set Descriptor

In deleting a file page set 140 descriptor, and SKCT 152 lock is needed, since no lock is issued on page set 140 for a partitioned file at allocation time. The locks must be held for at least commit duration, except the lock on the SKCT 152 of the current transaction, which must be held for deallocation duration. No page set 140 lock is needed since the SKCT 152 lock will prevent definitional change from impacting SQL 90 processing.

The procedure is:

1) Lock the target DBD 77 in exclusive mode with commit duration.

2) Lock the plans 156, 158 on the associated file 152 in exclusive mode with commit or deallocation duration.

3) Invalidate all PSCBs 230 on the target page set 140.

4) Drop the target file page set OBD 124 from DBD 77.

Altering a File Page Set Descriptor

In altering a file page set 140 descriptor, the page set open lock will serialize the open/close of the page set 140 so that no other transaction can retrieve the information contained in OBD 124 while a transaction is changing it.

The procedure is:

1) Lock the target DBD 77 in exclusive mode with commit duration.

2) Lock the open attribute of the associated file page set 140 in exclusive mode with commit duration.

3) Alter the target file page set 140 OBD 124 in the DBD 77.

Creating an Index Page Set Descriptor

In creating an index page set 140 descriptor, an SKCT 152 lock is used to guarantee the consistency of the index page set chain during DBMS 70 allocation of CUBs 220 for the given index and also during SQL updating and index maintenance. The locks must be held for at least commit duration, except the lock on the SKCT 152 of the current transaction, which must be held for deallocation duration.

The procedure is:

1) Lock the target DBD 77 in exclusive mode with commit duration.

2) Lock the plans 156, 158 on the associated record type 142 in exclusive mode with commit or deallocation duration.

3) Build the target index page set OBD 124 in DBD 77.

Deleting an Index Page Set Descriptor

In deleting an index page set 140 descriptor, an SKCT 152 lock is used to guarantee the consistency of the index page set chain during DBMS 70 allocation of CUBs 220 for the given index 144 and also during SQL 90 updating and index 144 maintenance. The locks are held for at least commit duration, except the lock on the SKCT 152 of the current transaction, which is held for deallocation duration.

The procedure is:

1) Lock the target DBD 77 in exclusive mode with commit duration.

2) Lock the plans 156, 158 on the associated record type 142 in exclusive mode with commit or deallocation duration.

3) Invalidate all PSCBs 230 on the target page set.

4) Drop the target index page set OBD 124 from the DBD 77.

Altering an Index Page Set Descriptor

In altering an index page set 140 descriptor, a page set open lock serializes the open/close of the page set so that no other transaction can retrieve information from an OBD 124 while a transaction is changing it.

The procedure is:

1) Lock the target DBD 77 in exclusive mode with commit duration.

2) Lock the open attribute of the target index page set 140 in exclusive mode with commit duration.

3) Alter the target index page set OBD 124 in DBD 77.

0100821

Claims:

1. A method for managing a data base in a computing apparatus referencing a relational database having a plurality of database objects and their descriptors, characterized by the steps of:

prebinding one or more transactions (101, 103, 153) to reference certain descriptors (77); and

during a concurrent definitional process, processing a current descriptor while maintaining the availability of data (76) and descriptors not impacted by the current definitional process.

2. The method of claim 1 wherein said plurality of database objects and their descriptors are stored in a logical segment, the method providing for online definition of descriptors while maintaining the availability of data and descriptors not impacted by a concurrent definitional process and including the further steps of:

determining the domain effected by the definitional process;

marking invalid to force rebinding of prebound transactions which reference descriptors within the domain effected by the definitional process;

inhibiting prior-to-commit access by transactions which have not been prebound to all objects within the domain effected by the definitional process: and

thereafter allowing all accesses to objects in the domain;

whereby descriptors outside of the domain are available to all transactions, and read only access is available to pre-bound transactions referencing descriptors within the domain which are not modified by this concurrent definitional proces

3. The method of claim 1 comprising the further steps of binding a target application plan according to the steps of:

locking a catalog (78) table page in exclusive mode with commit duration;

locking the target plan (150) in exclusive mode with commit duration;

producing target plan sections (210, 240); and

storing the target plan.

4. The method of claim 3 comprising the further steps of executing statement prebound within a plan according to the steps of:

locking the associated plan (150) in shared mode with commit duration;

locking the page set or page which is the target of the statement in shared or exclusive or update mode with commit duration; and

executing the statement (100, 102).

5. The method of claim 1 comprising the further step of executing a dynamic statement according to the steps of:

locking a target database descriptor (77) describing a page set or page which is the target of the statement in shared mode with commit duration;

locking the target page set or page (76) selectively in shared or exclusive or update mode with commit duration; and

executing the statement (100, 102).

6. The method of claim 1, comprising the further step of executing a utility job according to the steps of:

locking the target database descriptor (77) in exclusive mode with commit duration;

setting an utility-in-process flag in the target database descriptor;

locking a target page set or page (76) selectively in shared or exclusive or update mode with commit duration; and

doing the utility process.

7. The method of claim 1 comprising the further step of creating a new database descriptor according to the steps of:

locking the database identifier generator in exclusive mode with commit duration;

allocating a database identifier to the new database descriptor (77); and

building the new database descriptor.

8. The method of claim 7 comprising the further step of deleting a target database descriptor according to the steps of:

lock the target database descriptor (77) in exclusive mode with deallocation duration;

locking the plans (150) which reference the target database descriptor in exclusive mode with commit or deallocation duration;

deleting the plans referencing all files referenced by the target database descriptor;

invalidating all record type or index cursor blocks (220) and all page set cursor blocks (230) referencing any objects in the target database descriptor; and

dropping the target database descriptor.

9. The method of claim 1 comprising the further step of creating a file descriptor according to the steps of:

locking a target database descriptor (77) in which the file descriptor is to be created in exclusive mode with commit duration; and

building the file descriptor (124) in the target database descriptor.

10. The method of claim 9 comprising the further step of deleting a target file descriptor according to the steps of:

locking the target database descriptor (77) containing the target file descriptor in exclusive mode with commit duration;

locking all target plans (156, 158) referencing the target file (142) described by the target file descriptor to be deleted in exclusive mode with commit or deallocation duration;

deleting all plans (156, 158) referencing the target file;

invalidating all record type or index cursor blocks (220) referencing the target file; and

dropping the target file descriptor (126) from the target database descriptor.

11. The method of claim 10 comprising the further step of altering a target file descriptor according to the steps of:

locking the database descriptor (77) containing the target file descriptor in exclusive mode with commit duration;

locking the page set (76) associated with the file (140) described by the target file descriptor (124) in exclusive mode with commit duration; and

altering the target file descriptor (124).

12. The method of claim 1 comprising the further step of creating a record type descriptor according to the steps of:

locking the database descriptor (77) to contain the record type descriptor in exclusive mode with commit duration; and

building the record type descriptor (128).

13. The method of claim 12 comprising the further step of deleting a record type descriptor according to the steps of:

locking the database descriptor (77) containing the record type descriptor to be deleted in exclusive mode with commit duration;

locking target plans (156, 158) referencing the record type described by the record type descriptor to be deleted in exclusive mode with commit or deallocation duration;

deleting all target plans;

invalidating the record type (142) and all index cursor blocks (220) referencing the record type described by the record type descriptor (126) to be deleted; and

dropping the record type descriptor (126) being deleted from its database descriptor.

14. The method of claim 12 comprising the further step of altering a target record type descriptor according to the steps of:

locking the database descriptor containing the target record type descriptor in exclusive mode with commit duration;

locking plans (156, 158) referencing the target record type descriptor (126) in exclusive mode with commit or deallocation duration;

invalidating the record type (142) and all index cursor blocks (220) referencing the target record type descriptor; and

altering the target record type descriptor.

15. The method of claim 1 comprising the further step of creating an index descriptor according to the steps of:

locking the database descriptor (77) to contain the index descriptor in exclusive mode with commit duration;

locking the page set (76) associated with the file to be indexed in the index (140) to be described by the index descriptor being created in exclusive mode with commit duration; and

building the index descriptor (124) being created.

16. The method of claim 15 comprising the further step of deleting a target index descriptor according to the steps of:

locking the database descriptor (77) from which the target index descriptor (124) is to be deleted in exclusive mode with commit duration;

locking all plans (152) referencing the target index in exclusive mode with commit or deallocation duration;

deleting all plans (152) referencing the target index descriptor;

locking the page set (76) associated with the file indexed by the index described by the target index descriptor in exclusive mode with commit duration;

invalidating all index cursor blocks (220) referencing the target index descriptor; and

dropping the target index descriptor (124).

17. The method of claim 15 comprising the further step of altering a target index (140) descriptor (124) according to the steps of:

locking the database descriptor (77) containing the target index descriptor in exclusive mode with commit duration;

locking the page set (144) associated with the file indexed by the index described by the target index descriptor in exclusive mode with commit duration; and

altering the target index descriptor (124).

18. The method of claim 1 comprising the further step of creating a file page set (140) descriptor according to the steps of:

locking the database descriptor (77) to contain the file page set descriptor in exclusive mode with commit duration;

locking the plans (150) referencing the file (144) associated with the file page set (140) to be described in exclusive mode with commit or deallocation duration; and

building the file page set descriptor (124) being created in the database descriptor.

19. The method of claim 18 comprising the further step of deleting a file page set descriptor according to the steps of:

locking the database descriptor (77) containing the file page set descriptor (124) to be deleted in exclusive mode with commit duration;

locking the plans referencing the file page set descriptor to be deleted in exclusive mode with commit or deallocation duration;

invalidating all page set cursor blocks (230) referencing the file page set (140) descriptor being deleted; and

dropping the file page set descriptor (124) from its database descriptor.

20. The method of claim 18 comprising the further step of altering a page set (140) descriptor according to the steps of:

locking the database descriptor (77) containing the page set descriptor to be altered in exclusive mode with commit duration;

locking the open attribute of the file page set associated with the page set descriptor to be altered in exclusive mode with commit duration; and

altering the page set descriptor (124).

21. The method of claim 1 comprising the further step of creating an index page set (140) descriptor according to the steps of:

locking the database descriptor (77) to contain the index page set descriptor being created in exclusive mode with commit duration;

locking the plans (156, 158) referencing the record type (142) associated with the index page set to be described by the index page set descriptor being created in exclusive mode with commit or deallocation duration; and

building the index page set descriptor (124) in the database descriptor.

22. The method of claim 21 comprising the further step of deleting an index page set (140) descriptor according to the steps of:

locking the database descriptor (77) containing the index page set descriptor to be deleted in exclusive mode with commit duration;

locking the plans (156, 158) referencing record types (142) associated with the index page set descriptor being deleted in exclusive mode with commit or deallocation duration;

invalidating all page set cursor blocks (230) referencing the page set indexed by the index described in the index page set descriptor being deleted; and

dropping the index page set descriptor (124) being deleted from its database descriptor.

23. The method of claim 21 comprising the further step of altering an index page set (140) descriptor according to the steps of:

locking the database descriptor (77) containing the index page set descriptor to be altered in exclusive mode with commit duration;

locking the open attribute of the index page set described by the index page set (140) descriptor to be altered in exclusive mode with commit duration; and

altering the index page set descriptor (124).

24. A computing apparatus for managing a database including means for executing user transactions and a database storage means for storing a relational database having a plurality of database objects and their descriptors referenced by the transactions characterized by,

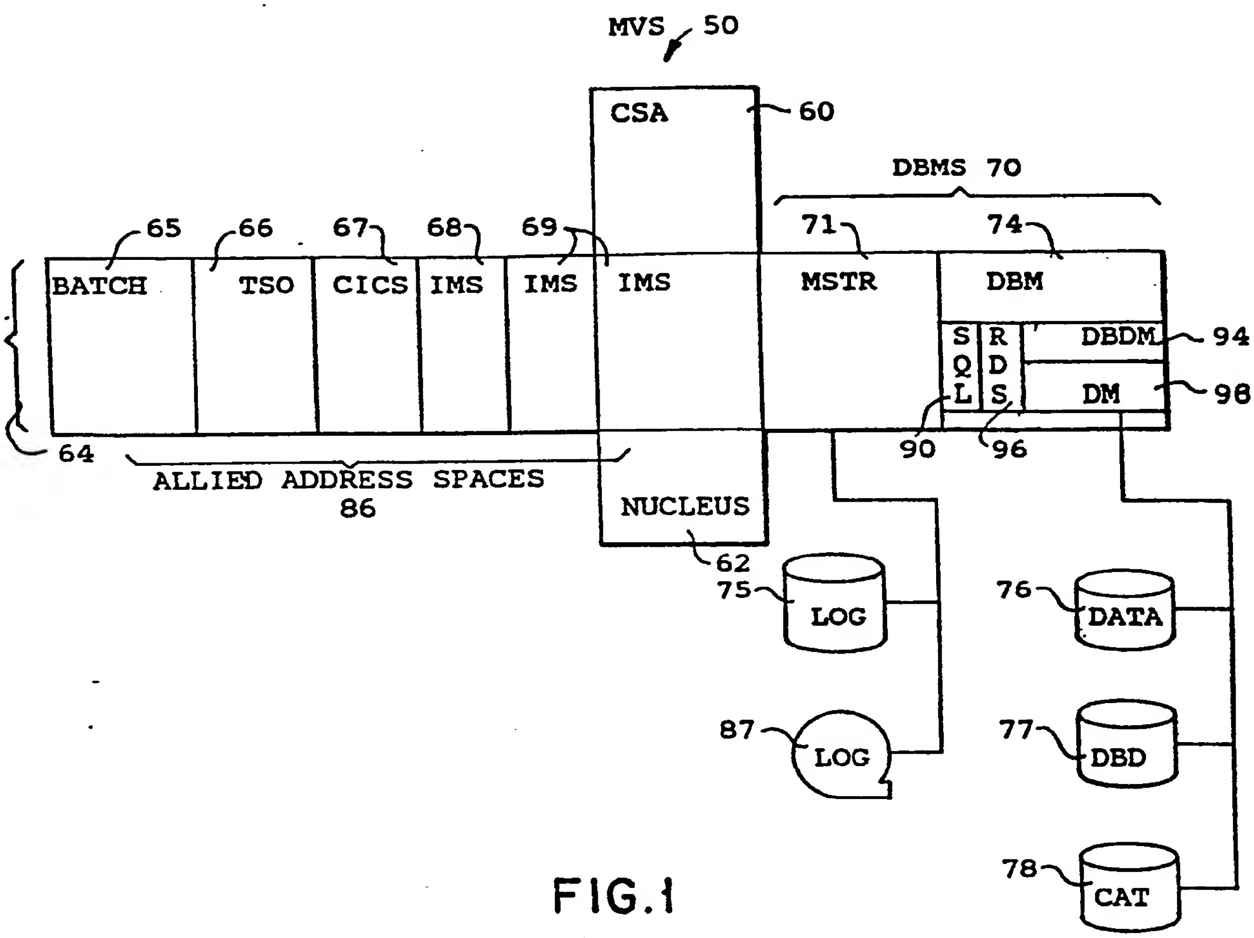
means for prebinding one or more transactions (101, 103, 153) to reference certain descriptors (77); and

means for processing during a current definitional process a database object descriptor while maintaining the availability of data (76) and descriptors not impacted by the current definitional process.

25. The computing apparatus of claim 24 including means for determing the domain effected by the definitional process,

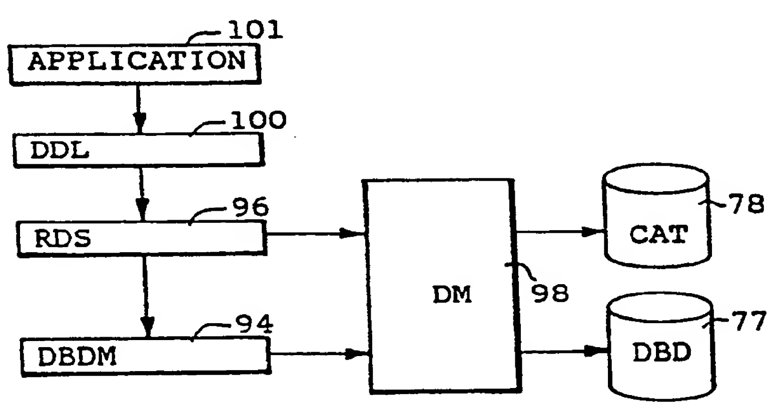
means for forcing rebinding of prebound transactions which reference descriptors within the domain effected by the definitional process; and

means for inhibiting prior-to-commit access by transactions which have not been prebound to all objects within the domain effected by the definitional process whereby the apparatus includes means for providing for online definition of descriptors while maintaining the availability of data and descriptors not impacted by a current definitional process.



MVS 50
CSA
60
DBMS 70
65
66
67
68
69
71
74
BATCH
TSO
CICS
IMS
IMS
IMS
MSTR
DBM
SQL
RDS
DBDM
94
DM
98
64
90
96
ALLIED ADDRESS SPACES
86
NUCLEUS
62
75
LOG
76
DATA
87
LOG
77
DBD
78
CAT

FIG.1



101
APPLICATION
100
DDL
96
RDS
94
DBDM
DM
98
CAT
78
DBD
77

FIG.2

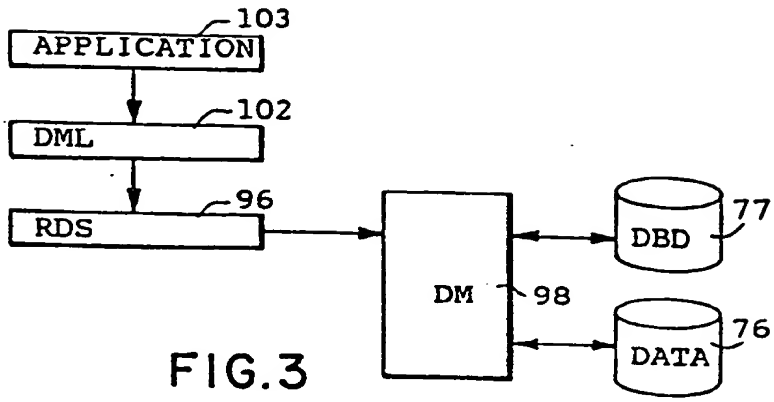


FIG.3

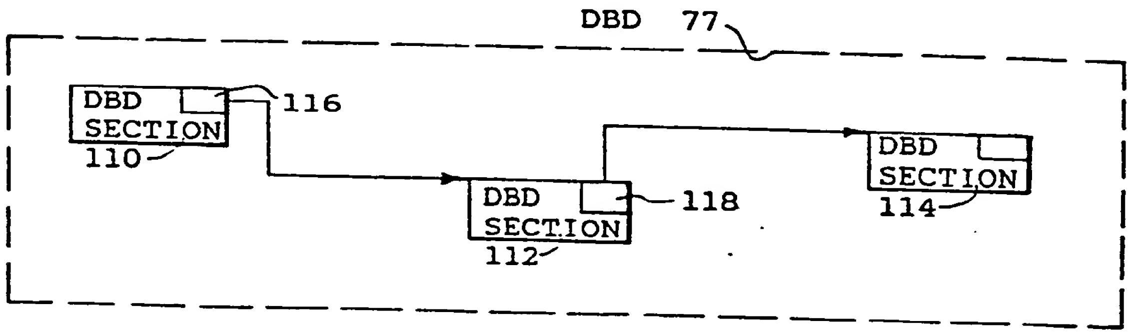


FIG.4

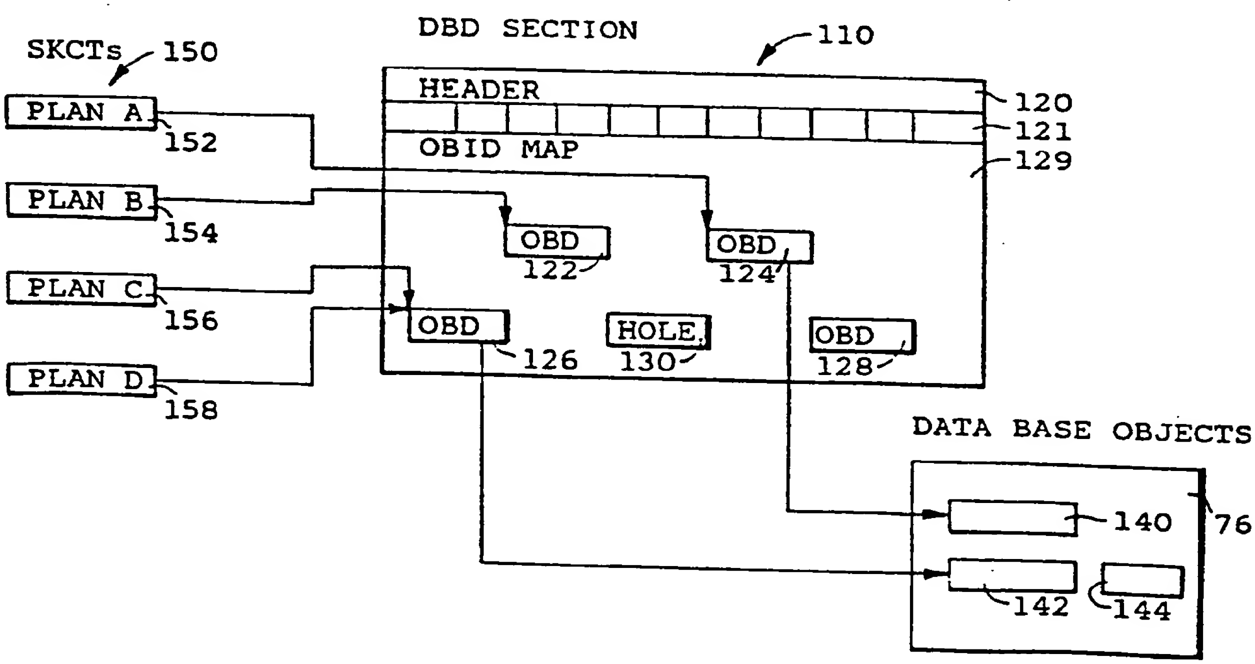


FIG.5

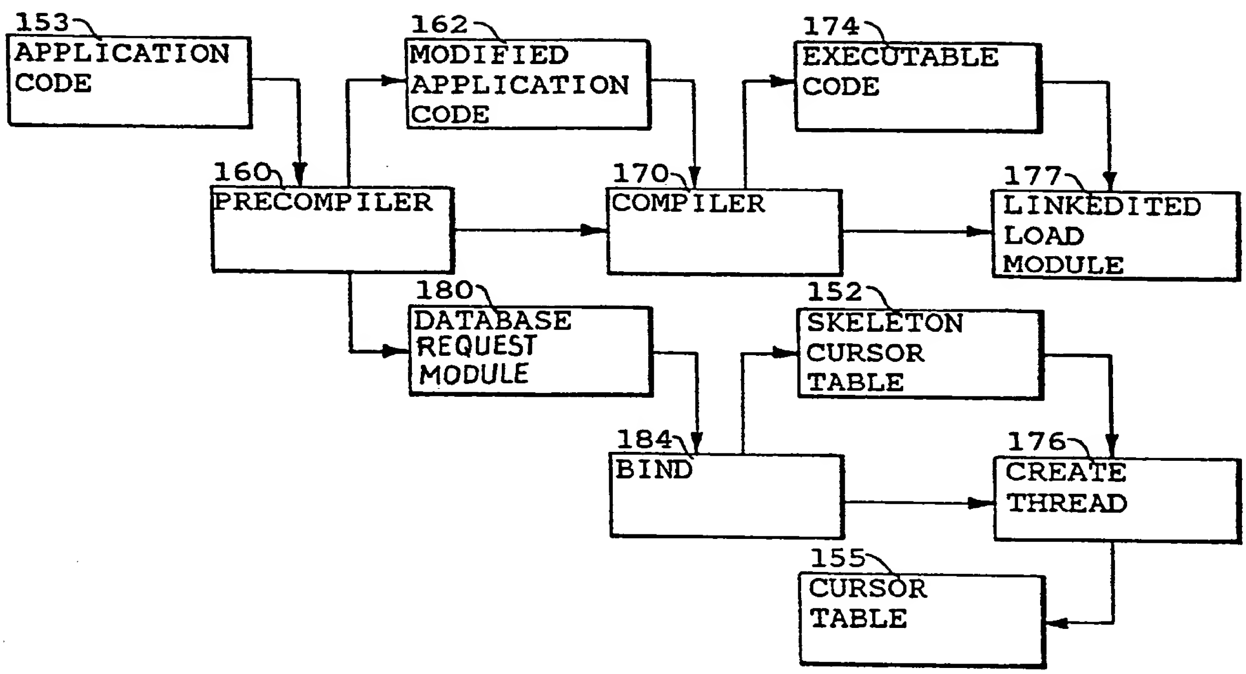

153
APPLICATION CODE
162
MODIFIED APPLICATION CODE
174
EXECUTABLE CODE
160
PRECOMPILER
170
COMPILER
177
LINKEDITED LOAD MODULE
180
DATABASE REQUEST MODULE
152
SKELETON CURSOR TABLE
184
BIND
176
CREATE THREAD
155
CURSOR TABLE

FIG.6

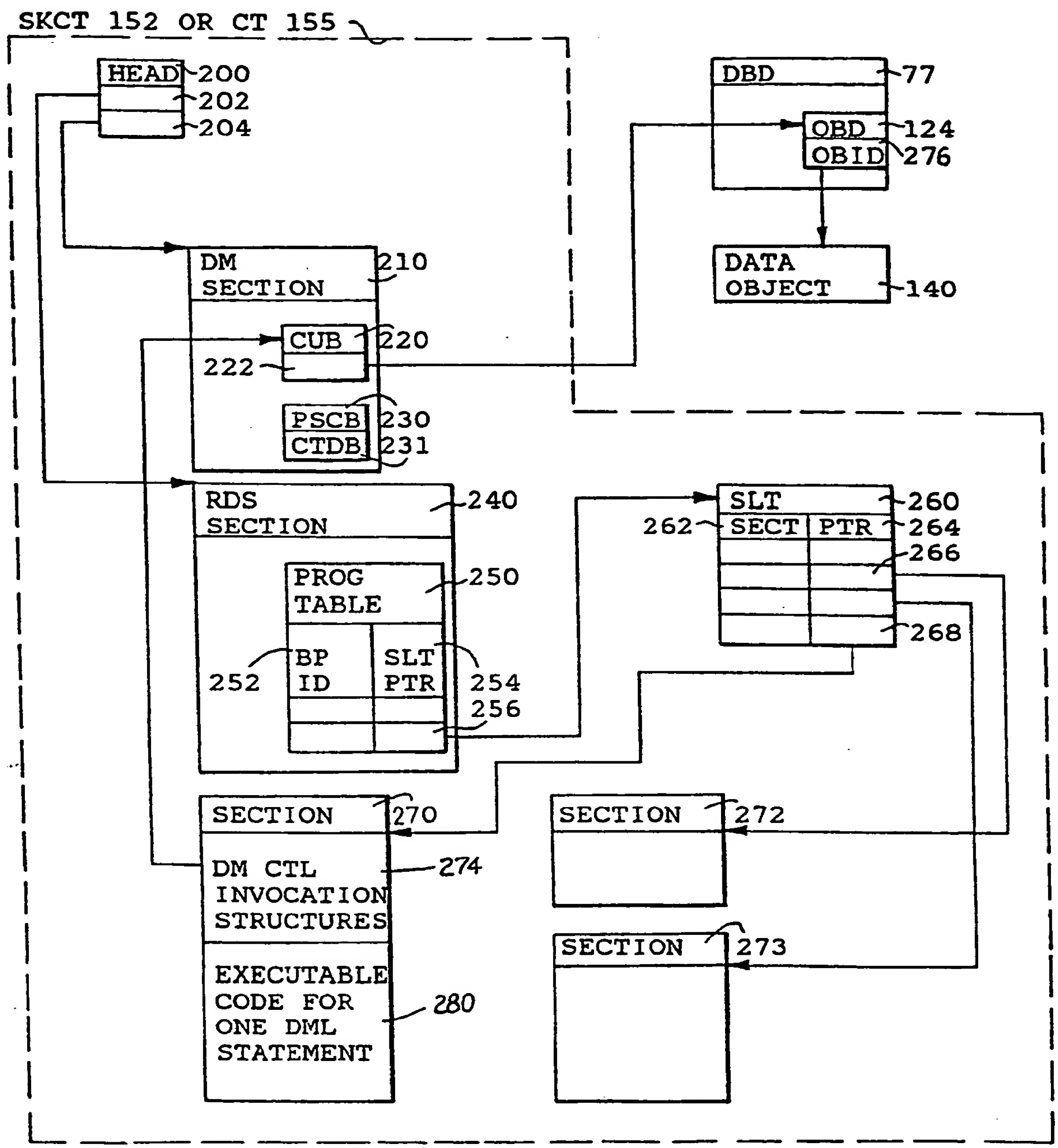

SKCT 152 OR CT 155
HEAD
200
202
204
DBD
77
OBD
124
OBID
276
DATA OBJECT
140
DM SECTION
210
CUB
220
222
PSCB
230
CTDB
231
RDS SECTION
240
PROG TABLE
250
BP ID
252
SLT PTR
254
256
SLT
260
262
SECT
PTR
264
266
268
SECTION
270
DM CTL INVOCATION STRUCTURES
274
EXECUTABLE CODE FOR ONE DML STATEMENT
280
SECTION
272
SECTION
273

FIG. 7

THIS PAGE BLANK (USPTO)